THEORY AND PRACTICE OF HIERARCHICAL CONTROL

1 November 1980

This work was partially funded by the
Manufacturing Technology Division,
Air Force Material Laboratory,
Air Force Systems Command,
Wright Patterson Air Force Base, Ohio 45433

James S. Albus
Anthony J. Barbera
Roger N. Nagel
A123, Metrology Bldg.
Programmable Automation
National Bureau of Standards
Washington, DC 20234
(301) 921-2381

INTRODUCTION

The problem of controlling a sensory-interactive robot is similar in many respects to that of controlling any complex system such as an army, a government, a business, or a biological organism. The command and control structure for such systems is invariably a hierarchy wherein goals, or tasks, selected at the highest level are decomposed into sequences of subtasks which are passed to the next lower level in the hierarchy. This same procedure is repeated at each level until at the bottom of the hierarchy there is generated a sequence of primative tasks which can be executed with single actions. Sensory feedback enters the hierarchy at many different levels to alter the task decomposition so as to accomplish the highest level goal in spite of uncertainties or unexpected conditions in the environment.

Problem reduction, or task decomposition, is a classic hierarchical approach to problem solving [1]. In Figure 1 the problem A may be solved either by solving subproblems B or C. The nodes B and C are called OR nodes. Problem B has been decomposed into two subproblems D and E, both of which must be solved in order to solve B. Similarly problem C is decomposed into two subproblems F and G both of which must be solved to solve C. Nodes D and E and nodes F and G are called AND nodes and are shown with a bar joining their arcs. At some point in the decomposition process, primitive subproblems (ones whose solutions are known) are generated. In Figure 1, if either D and E, or F and G are primitive subproblems, then the problem A is solved. A structure like that shown in Figure 1 is called an AND/OR graph.

PLANNING VS. CONTROL

Most research with problem reduction and AND/OR graphs has been done in the study of planning, not the study of control systems. In planning the emphasis is on mechanisms for searching AND/OR graphs to find, and hopefully to optimize solutions. This is relevant to problems in abstract mathematics or to board games where there can be a planning period between each move.

In control systems, however, time is a critical variable. Action is continuous and on-going. If planning is to be done, it must be done off-line, or in a background mode because in real-time control systems, searching and backtracking are unacceptable, and time delays between observing and acting are a major source of instability.

Nevertheless, AND/OR decomposition is a valuable tool for analyzing hierarchical networks. What is needed to apply AND/OR graphs to hierarchical control systems is a mechanism for generalizing

from a discrete to a continuous form with time explicitly represented.

AND/OR TRAJECTORIES

As part of our research program at NBS [2] we have developed a notation for continuous time-dependent AND/OR graphs which essentially merges sequences of AND nodes into smooth trajectories and lets OR nodes be capable of lying anywhere within large regions of multidimensional space. In order to visualize this concept let us define a task decomposition operator as a continuous single valued function H which transforms an input vector $\underline{S} = (s_1, s_2, \ldots s_N)$ composed of continuous time-dependent variables s_i, into an output vector $\underline{P} = (p_1, p_2, \ldots p_L)$ composed of continuous time dependent variables p_j.

H thus maps each input vector $\underline{S}$ in input space into an output vector $\underline{P}$ in output space. If the input vector moves as a function of time (as a result of any of its components changing with time) then $\underline{S}$ will describe a trajectory T_S through input space. Assume the function H samples the input periodically and after a short computation delay produces an output. Thus as $\underline{S}$ moves along T_S, P will move along T_P. This is illustrated in Figure 2.

We can now divide the input vector into two parts.

$\underline{C} = (s_1, \ldots s_i)$ - where we can call $\underline{C}$ some command vector, and
$\underline{F} = (s_{i+1}, \ldots s_N)$ - where we can call $\underline{F}$ some feedback vector such that $\underline{S} = \underline{C} + \underline{F}$ as shown in Figure 3.

Assume for the moment that $\underline{C}$ (a command such as telling the robot to locate and pick up an object on a moving conveyor) is held constant while $\underline{F}$ (feedback from sensors such as an encoder that tells the position of the conveyor and from lower level vision and touch sensors) is allowed to vary. If $\underline{F}$ varies smoothly, the vector $\underline{S}$ will trace out the trajectory T_S as shown in Figure 4. The function H maps each point on T_S into a point on T_P. (H might compute the correct cartesian coordinate values T_P for the robot as the input T_S varies in time.)

The tip of the vector $\underline{C}$ now corresponds to the selection of an OR node. The tip of the vector $\underline{F}$ traces out a series of AND nodes under this OR node. In the case where $\underline{F}$ varies smoothly the AND nodes merge into a continuous trajectory. If $\underline{F}$ moves in steps from $\underline{F}^1$ to $\underline{F}^2$ to $\underline{F}^3$, then the vector $\underline{S}$ jumps from one discrete point along T_S to another. Thus the continuous analysis degenerates to the discrete when the inputs are discrete.

The variation in $\underline{F}$ may be caused by external forces imposed by the environment, or by actions produced by the output, or both. One or more of the variables in the feedback vector $\underline{F}$ may even be taken directly from the output vector $\underline{P}$. In the latter case, the H operator becomes the transition function for a state automaton. In any of these cases the result is that a single command vector $\underline{C}$ produces a sequence of output vectors T_p. The process is driven by the sequence of feedback vectors $\underline{F}^1$, $\underline{F}^2$, $\underline{F}^3$, ... The superscript $\underline{F}^k$ denotes the vector $\underline{F}$ at time k .

The sequence of operations illustrated in Figure 4 can also be viewed as a decomposition of a command $\underline{C}$ into a sequence of subcommands $\underline{P}^1$, $\underline{P}^2$, $\underline{P}^3$. The vector $\underline{C}$ may, of course, be a symbol standing for any number of things such as a task, a goal, or a plan. In such cases the output string $\underline{P}^1$, $\underline{P}^2$, $\underline{P}^3$, represents a sequence of subtasks, subgoals, or sub-plans respectively.

Figure 4, can also be viewed as a servomechanism where $\underline{C}$ defines a setpoint and $\underline{F}$ provides the feedback which is used to compute an error signal. H then is the servo system transform function. In this case there are many practical problems concerned with stability, speed, gain, delay, phase shift, etc. In our notation these are all embeded in the H functions. If the H functions are correctly formulated and defined over the entire space traversed by the $\underline{S}$ input, then the output T_p will drive the physical actuators in such a way that the goal is achieved, i.e., the error between the command $\underline{C}$ and the results $\underline{P}$ is nulled, and stability is maintained under all conditions.

HIERARCHICAL CONTROL

Assume that the command vector $\underline{C}$ in Figure 4 changes such that it steps along the trajectory T_C as shown in Figure 5. The result is that the sequence of input commands C^{1-3}, C^{4-5}, ... produce the sequence of output vectors $\underline{P}^1$, $\underline{P}^2$, $\underline{P}^3$, $\underline{P}^4$, $\underline{P}^5$, In this case the sub-sequence $\underline{P}^1$, $\underline{P}^2$, $\underline{P}^3$, is called by the command $\underline{C}^{1-3}$ and driven by the feedback $\underline{F}^1$, $\underline{F}^2$, $\underline{F}^3$. The sub-sequence $\underline{P}^4$, $\underline{P}^5$ is called by $\underline{C}^{4-5}$ and driven by $\underline{F}^4$, $\underline{F}^5$, etc.

If we now represent time explicitly, the C, F, and P vectors and trajectories of Figure 5 appear as shown in Figure 6. The fact that C remains constant while the feedback changes from $\underline{F}^1$, to $\underline{F}^2$ to $\underline{F}^3$ means that the trajectory T_c is parallel to the time axis over that interval. The jump from C^{1-3} to C^{4-5} causes an abrupt shift in the T_c trajectory in the time interval between $\underline{F}^3$ and $\underline{F}^4$. Note that each instant of time can be represented by a plane (or set of co-planar regions) perpendicular to the time axis. Each plane contains a point from each trajectory and represents a snapshot of all the vectors simultaneously at a specific instant in time.

We are now ready to consider a hierarchy of servomechanisms, or task decomposition operators, as is shown on the left side of Figure 7. Here the highest level input command $\underline{C}_4$ is a symbolic vector denoting the complex task (ASSEMBLE AB). The subscript $\underline{C}_k$ denotes the $\underline{C}$ vector at the k-th level in the hierarchy.

Note that in Figure 7, vectors are not repeatably drawn for each instant of time during the trajectory segments when they are reasonably constant. Thus, $\underline{C}_4$ is shown only at the beginning and end of the trajectory segment labeled (ASSEMBLE AB). $\underline{C}_2$ is shown only at the transition points between (REACH to A), (GRASP), (MOVE TO X), (RELEASE), etc. It should be kept in mind, however, that H_4 computes $\underline{P}_4$ continuously and produces an output at every instant of time just as H_2 does $\underline{P}_2$.

The feedback $\underline{F}_4$ may contain highly processed visual scene analysis data which identifies the general layout of the work space and thereby determines which output vectors $\underline{P}_4$, and hence which simple task commands $\underline{C}_3$, should be selected and in which order. $\underline{F}_4$ may also contain data from $\underline{P}_4$ and $\underline{P}_3$ which indicate the state of completion of the decomposition of $\underline{C}_4$. $\underline{F}_4$ combines with $\underline{C}_4$ to define the complete input vector $\underline{S}_4$. The H_4 operator produces an output vector $\underline{P}_4 = H_4\ (\underline{S}_4)$. As the feedback $\underline{F}_4$ varies with time, the input vector $\underline{S}_4$, and hence the output vector $\underline{P}_4$, moves along a trajectory generating a sequence of "simple task" commands at $\underline{C}_3$ such as (FETCH A), (FETCH B), (MATE B TO A), (FASTEN B TO A), etc. as shown in Figure 7.

C_3 is also a symbolic vector which identifies one of a library of "simple task" commands together with the necessary modifiers and arguments. Feedback at $\underline{F}_3$ may identify the position and orientation of the parts A and B and also caarry state sequencing information from outputs $\underline{P}_3$ and $\underline{P}_2$ As $\underline{F}_3$ varies with time it drives the input $\underline{S}_3$, and hence $\underline{P}_3$, along a trajectory generating a sequence of "elemental movement" commands at $\underline{C}_2$ such as (REACH TO A) (GRASP), (MOVE TO C), (RELEASED), etc.

Feedback at $\underline{F}_2$ may contain information from proximity sensors or force and touch sensors indicating the fine positioning error between the fingers and the objects to be manipulated together with state sequencing information derived from $\underline{P}_2$ and $\underline{P}_1$. The operator H_2 produces $\underline{P}_2$ which denotes the proper velocity vectors C_1 for the manipulator hand in joint angle coordinates. Feedback $\underline{F}_2$ also provides joint angle position data necessary for the coordinate transformations performed by H_2 . $\underline{P}_2$ provides reference, or set point commands $\underline{C}_1$ to the servomechanism operator H_1. $\underline{F}_1$ provides position, velocity, and force information for the traditional servo computations. The output $\underline{P}_1$ is a set of drive signals to the actuators.

Feedback enters this hierarchy at every level. At the lowest levels the feedback is unprocessed, or nearly so, and hence is fast acting with very short loop delays. At higher levels feedback data passes through more and more stages of an ascending sensory-processing hierarchy. Feedback thus closes a real time control loop at each level in the hierarchy. The lower level loops are simple and fast acting. The higher level loops are more sophisticated and slower.

At each level the feedback vector $\underline{F}$ drives the output vector $\underline{P}$ along its trajectory. Thus, at each level of the hierarchy, the time rate of change of the output vector $\underline{P}$ will be of the same order of magnitude as the feedback vector $\underline{F}$, and considerably more rapid than the command vector $\underline{C}$. The result is that each stage of the behavior generating hierarchy effectively decomposes an input task represented by a slowly changing $\underline{C}$ into a string of subtasks represented by a more rapidly changing $\underline{P}$.

At this point we perhaps should emphasize that the difference in time rate of change of the vectors at various levels in the hierarchy does not imply that the H operators are computing slower at the higher levels than at the lower. We will, in fact, assume that every H operator transforms S into P with the same computational delay Δt at all levels of the hierarchy. That is

$$\underline{P}_i\ (t) = H_i\ (\underline{S}_i\ (t-\Delta t)) \text{ or } \underline{P}_i^k = H_i\ (\underline{S}_i^{k-1})$$

at every level. The slower time rate of change of $\underline{P}$ vectors at the higher levels stems from the fact that the $\underline{F}$ vectors driving the higher levels convey information about events which occur less frequently, and in some cases components of higher level $\underline{F}$ vectors may require the integration of information over long time intervals or the recognition of symbolic messages with long word lengths.

When we represent time explicitly as in Figure 7, we can label the relatively straight segments of the T_C trajectories as tasks and subtasks. Transitions between the subtasks in a sequence correspond to abrupt changes in T_C.

If we do not represent time explicitly, the relatively constant $\underline{C}$ vectors correspond to nodes as in Figure 5. The resulting tree structure represents a classical AND/OR decomposition of a task into sequence of subtasks where the discrete $\underline{C}$ vectors correspond to OR nodes and the rapidly changing sequences of $\underline{P}$ vectors become sets of AND nodes under those OR nodes.

INTENTIONAL OR PURPOSIVE BEHAVIOR

Figure 7 illustrates the power of a hierarchy of multivariant servos to generate a lengthy sequence of behavior which is both goal-directed and appropriate to the environment. Such behavior appears to an external observer to be intentional, or purposive. The top level input command is a goal, or task, which is successively decomposed into subgoals, or subtasks, at each stage of the control hierarchy until at the lowest level output signals drive the muscles, or other actuators, producing observable behavior.

To the extent that the $\underline{F}$ vectors at the various levels contain sensory information from the environment, the task decompositions at those levels will be capable of responding to the environment. What the response to each $\underline{F}$ vector is depends on the H function at that level. If the $\underline{F}$ vector at any level is made up solely of internal variables, then the decomposition at that level will be stereotyped and insensitive to conditions in the environment.

Whether or not the hierarchy is driven by external, or internal variables, or both, the highest level input command commits the entire structure to an organized and coordinated sequence of actions which under normal conditions will achieve the goal or accomplish the task. The selection of a high level input command in a biological organism thus corresponds to an intent, or purpose, which depending on circumstances, may or may not be successfully achieved through the resulting hierarchical decomposition into action.

The success or failure of any particular task performance, or goal- seeking action depends upon whether the H functions at each level are capable of providing the correct mappings so as to maintain the output trajectory within a region of successful performance despite perturbations and uncertainties in the environment.

At all levels variations in the $\underline{F}$ vectors due to irregularities in the environment cause T_S trajectories to vary from one task performance to the next. This implies that while there may exist a set of ideal trajectories through S and P space at each level of the hierarchy corresponding to an ideal task performance, there also must be an envelop of close-to-ideal trajectories which correspond to successful, but not perfect, task performance. This is illustrated in Figure 8.

The H functions must not only be defined along the T_S trajectories corresponding to ideal performance, but also in the regions around the ideal performance, so that any deviation from the

ideal is treated as an error signal which generates an action designed to restore the actual trajectory to the ideal, or at least to maintain it within the region of successful performance.

Small perturbations can usually be corrected by low level feedback loops as shown in Figure 9. These involve relatively little sensory data processing, and hence are fast acting. Larger perturbations in the environment may overwhelm the lower level feedback loops and require strategy changes at higher levels in order to maintain the system within the region of successful performance. This is illustrated in Figure 10. Major changes in the environment are detected at higher levels after processing through several levels of pattern recognizers. This produces differences in the $\underline{F}$ vectors at the higher level which in turn produces different $\underline{C}$ vectors to lower levels. The result is an alternative higher level strategy to cope with the perturbation.

Of course, if the H functions do not provide stability, or if the environment is so perverse that the system is overwhelmed, then the trajectories diverge from the region of successful performance and failure occurs.

Note that Figure 7 illustrates only a single specific performance of a particular task. None of the alternative trajectories which might have occurred under different circumstances with a different set of $\underline{F}$ vectors are indicated. These alternatives which might have occurred can be illustrated in the plane orthogonal to the time axis.

In Figure 11, we use an example from a biological goal-seeking control system; that of a fish. This figure illustrates the set of alternative $\underline{C}$ vectors postulated by Tinbergen [4] to reside at various levels in the behavior-generating hierarchy of the male three-spined stickleback fish. This Figure represents a snap shot, or single cut through space orthognal to the time axis. $\underline{C_6}$, the highest level goal is survival. The feedback $\underline{F_6}$ consists of variables indicating water temperature and depth, blood chemistry, and hormone levels generated by length-of-day detectors. When the hormone levels indicate the proper time of year, and the blood chemistry does not call for feeding behavior, then migratory behavior will be selected until warm, shallow water is detected. The $\underline{F_6}$ vector will then trigger the reproduction subgoal. When $\underline{C_5}$ indicates (REPRODUCTION), the $\underline{F_5}$ vector indicating a red male in the territory will select the (FIGHT) command to $\underline{C_4}$. When $\underline{C_4}$ indicates (FIGHT) and the intruder threatens, a $\underline{C_3}$ will be selected, and so on. At each level, a different feedback vector would select a different lower level subgoal. For example, if $\underline{F_5}$, indicates a female in the territory, $\underline{C_4}$ will become (MATE), and the type of mating behavior selected will depend on $\underline{F_4}$.

In simple creatures like the stickle-back fish, the sensory stimuli that produce E vectors which trigger behavioral trajectories are called "innate releasing mechanisms." Innate releasing mechanisms and their associated behavioral patterns have been studied extensively in a number of insects such as the digger wasp and various species of bees and ants, several fish, and many birds, including the herring-gull, the turkey, and the golden eye drake [5].

In these relatively simple creatures, behavior is sufficiently sterotyped that it can be described in terms of a small set of behavioral patterns triggered by a equally small set of sensory stimuli. This suggests that insects, fish, and birds have only a few levels in their control hierarchies and a small set of behavior patterns stored as H functions at each level. It further implies there are few externally driven components in the F vectors at each level. Behavior trajectories are internally driven with only a few branch points controlled by sensory data processed through simple pattern recognizers. The trajectory segments driven entirely by internal variables are what are known as fixed action patterns, or tropisms. The external variables which control the relatively few branch points are the innate releasing mechanisms.

The types of goal-seeking behavior we might expect to obtain from industrial robots over the next decade or two is of the same general level of complexity as that of an insect or simple fish. Four or five levels in the control hierarchy with a library of ten or twelve control subroutines (C vectors) available at each level, and a reasonable number (less than ten) of branch points in each subroutine is more than adequate to generate extremely complex sensory-interactive, goal-directed behavior in the constrained environment of a shop floor.

Figure 12 illustrates a set of trajectories in which there is opportunity for branching at several different levels at every step along each trajectory. At each instant in time the C vector to any particular level depends upon what the C and F vectors were to the next higher level at the previous instant. Thus, a change in the F vector at any level causes an alternative C vector to be sent to the level below. Behavior is continuously modified at all levels by external variables and hence, does not appear stereotyped at all.

In general, the complexity of behavior which can be generated by a control hierarchy depends on four factors.

1. The number of levels in the control hierarchy,
2. The number of feedback variables which enter each level,
3. The sophistication of the H functions which reside at each level,
4. The sophistication of the sensory processing systems

which extract feedback variables for use by the various H functions.

SENSORY PROCESSING IN A CONTROL HIERARCHY

The fundamental problem of pattern recognition is to name the patterns. All the patterns with the same name are in the same class. When a pattern has been given a name we say it has been recognized. For example, when the image of a familiar face falls on my retina and I say "That's George," I have recognized the visual pattern by naming it.

Any spatial pattern can be represented as a vector. For example a picture can be represented as an array, or ordered list, of brightness or color values. A symbolic character can be represented as an ordered list of features (or arbitrary numbers as in the ASCII convention). Any temporal pattern can be represented as a trajectory through an N-dimensional space. For example, an audio pattern is a sequence of pressure or voltage values, i.e., a one dimensional trajectory. A moving picture of televison scene corresponds to a sequence of picture vectors, i.e., a N-dimensional trajectory where N is the number of picture resolution elements or pixels.

At this point we need to introduce some new notation so as to clearly distinguish between vectors in the sensory-processing hierarchy and those in the behavior-generating hierarchy. Thus we will define the input vector to a sensory processing module as

$\underline{D} = \underline{E} + \underline{R}$ where $\underline{E} = (d_1 , d_2 , \ldots , d_i)$ is a vector, or list, of data variables derived from sensory input from the external environment and

$\underline{R} = (d_{i+1} , \ldots , d_N)$ is a vector of data variables derived from recalled experiences, or internal context. The mapping operator in the sensory-processing hierarchy will be denoted G and the output $\underline{Q}$ such that

$\underline{Q} = G (\underline{D})$

We can now define a $\underline{D}$ vector to represent a sensory pattern plus context such that each component d_i represents a data point or feature of the pattern plus context. The existence of the $\underline{D}$ vector within a particular region of space therefore corresponds to the occurrence of a particular set of features or a particular pattern in a particular context. The recognition problem then is to find a G function which computes an output vector $\underline{Q} = G (\underline{D})$ such that $\underline{Q}$ is the name of the pattern and context $\underline{D}$ as shown in Figure 13.

In other words G can recognize the existence of a particular pattern and context (i.e., the existence of $\underline{D}$ in a particular region

of input space) by outputting the name $\underline{Q}$. For example,

$\underline{Q}$ = Class I whenever $\underline{D}$ is in Region 1

$\underline{Q}$ = Class II whenever $\underline{D}$ is in Region 2

etc.

In the case where the $\underline{D}$ vector is time dependent, an extended portion of a trajectory T_D may map into a single name $\underline{Q}$ as shown in Figure 14. It then is possible by integrating $\underline{Q}$ over time and thresholding the integral to detect, or recognize, a temporal pattern T_D such as a sound or a visual movement.

The recognition, or naming, of a temporal pattern as illustrated in Figure 14 is the inverse of the decomposition of a task as illustrated in Figures 4-7. In task decomposition a slowly varying command $\underline{C}$ is decomposed into a rapidly changing output $\underline{P}$. In pattern recognition a rapidly changing sensory experience $\underline{E}$ is recognized by a slowly varying name $\underline{Q}$.

THE USE OF CONTEXT

It frequently occurs in pattern recognition, or signal detection, that the instantaneous value of the sensory input vector $\underline{E}$ is ambiguous or misleading. This is particularly true in noisy environments or in situations where data dropouts are likely to occur. In such cases the ambiguity can often be resolved, or the missing data filled in if the context can be taken into account, or if the classification decision can make use of some additional knowledge or well founded prediction regarding what patterns are expected.

The addition of context or prediction variables $\underline{R}$ to the sensory input $\underline{E}$ such that $\underline{D} = \underline{E} + \underline{R}$ increases the dimensionality of the pattern input space. The context variables thus can shift the total input (pattern) vector $\underline{D}$ to different parts of input space depending on the context. Thus, as shown in Figure 15 the ambiguous patterns $\underline{E_1}$ and $\underline{E_2}$ which are too similar to be reliably recognized as in separate classes, can be distinguished when accompanied by context $\underline{R_1}$ and $\underline{R_2}$.

In robot control many variables can serve as context variables. In fact any information about anything occurring simultaneously with the input pattern can be regarded as context. Thus context can be data from other sensory modalities as well as information regarding what is happening in the behavior-generating hierarchy. In many cases, data from this latter source is particularly relevant to the pattern recognition task, because the sensory input at any instant of time depends heavily upon what action is currently being executed. For example in biological systems the

information from the behavior-generating hierarchy provides contextual information necessary for the visual processing hierarchy to distinguish between motion of the eyes and motion of the room about the eyes.

In a classic experiment, von Holst and Mittelstaedt [6] demonstrated that this kind of contextual data pathway actually exists in insects. They observed that a fly placed in a chamber with rotating walls will tend to turn in the direction of rotation so as to null the visual motion. They then rotated the fly's head 180 degrees around its body axis (a procedure which for some reason is not fatal to the fly) and observed that the fly now circled endlessly because by attempting to null the visual motion it was now actually increasing it. Later experiments with motion perception in humans showed that the perception of a stationary environment despite motion of the retinal image caused by moving the eyes is dependent on contextual information derived from the behavior-generating hierarchy. The fact that the context is actually derived from the behavior-generating hierarchy rather than from sensory feedback can be demonstrated by anesthetizing the eye muscles and observing that the effect depends on the intent to move the eyes, and not the physical act of movement. The perceptual correction occurs even when the eye muscles are paralyzed so that no motion actually results from the conscious intent to move.

THE INTERNAL WORLD MODEL

Contextual information can also provide predictions of what sensory data to expect. This allows the sensory-processing modules to do predictive filtering, to compare incoming data with predicted data, and to fly-wheel through noisy data or data dropouts.

The mechanism by which such predictions, or expectations, can be generated is illustrated in Figure 16. Here contextual input for the sensory-processing hierarchy is shown as being processed through an M module before being presented to the sensory pattern recognition G modules at each level. Input to the M modules derive from the $\underline{P}$ vector of the corresponding behavior-generating hierarchy at the same level as well as an $\underline{X}$ vector which includes context derived from other areas of the brain such as other sensory modalities or other behavior-generating hierarchies. These M modules compute $\underline{R} = M(\underline{P} + \underline{X})$. Their position in the links from the behavior-generating to the sensory-processing hierarchies allow them to function as a predictive memory. They are in a position to store and recall (or remember) sensory experiences ($\underline{E}$ vector trajectories) which occur simultaneous with $\underline{P}$ and $\underline{X}$ vector trajectories in the behavior-generating hierarchy and other locations within the brain. For example, data may be stored

in each M module by setting the desired output $\hat{R}$ equal to the sensory experience vector E. At each instant of time t = k sensory data represented by E^k will then be stored on the memory address defined by the $P^k + X^k$ vector. The result will be that the sensory experience represented by the sensory data trajectory T_E will be stored in association with the context trajectory T_{P+X}

Any time afterwards, t = k + j, a reoccurrence of the same context vector $P^{k+j} + X^{k+j} = P^k + X^k$ will produce an output R^{k+j} equal to the E^k stored at time t = k. Thus a reoccurrence of the same context trajectory T_{P+X} will produce a recall trajectory T_R equal to the earlier sensory experience T_E. These predictive memory modules thus provide the sensory-processing hierarchy with a memory trace of what sensory data occurred on previous occasions when the motor generating hierarchy (and other parts of the brain) were in similar states along similar trajectories. This provides the sensory-processing system with a prediction of what sensory data to expect. What is expected is whatever was experienced during similar activities in the past.

In the ideal case, the predictive memory modules M will generate an expected sensory data stream T_R which exactly duplicates the observed sensory data stream T_E. To the extent that this occurs in practice it enables the G modules to apply very powerful mathematical techniques to the sensory data. For example the G modules can use the expected data T to:

1. Perform cross-correlation or convolution algorithms to detect sync patterns and information bearing sequences buried in noise,
2. Flywheel through data dropouts and noise bursts,
3. Detect (or recognize) deviations or even omissions from an expected pattern as well as the occurrence of the pattern in its expected form.

If we assume, as shown in Figure 16 that predictive recall modules exist at all levels of the processing-generating hierarchy, then it is clear that the memory trace itself is multileveled. In order to recall an experience precisely at all levels, it is necessary to generate the same context (i.e., $P + X$ address) at all levels as existed when the experience was recorded.

We can say that the predictive memory modules M define an internal model of the external world. They provide answers to the question, "if I do such and so, what will happen?" The answer is that whatever happened before when such and so was done, will probably happen again. In short, IF I do Y, THEN Z will happen when Z is whatever was stored in predictive memory the last time (or some statistical average over the N last times) that I did Y, and Y is some action such as performing a task, or pursuing a goal in a particular environment or situation, which is

represented internally by the $\underline{P}$ vectors at the various different levels of the behavior-generating hierarchy and the $\underline{X}$ vectors describing the states of various other sensory-processing behavior-generating hierarchies.

There are three aspects of a control hierarchy: One is the organizational hierarchy, shown on the left side of Figure 17, which defines the command and control structure and specifies the relationship between the various processing and generating modules. Two is the computational hierarchy, shown at the center in Figure 17, which defines the state variables and the functions that map input variables into output variables. Three is the behavioral hierarchy which defines the flow of actions that give rise to observable behavior. The right side of Figure 17 is an illustration of a behavioral hierarchy. It should be noted that the organizational hierarchy is not a strict tree structure, as there is much information flow between computational modules at the same level, and even between levels. Nevertheless, the primary flow of command and control information is vertical along the chain of command, and not lateral between modules at the same level. The lines between the three hierarchies illustrate the relationships between the computational structure, the state variables, and the flow of time. This way of viewing the problem provides a mathematical formalism that is clear, concise, and computationally useful in both analysing and synthesizing intelligent control system performance.

SOFTWARE IMPLEMENTATION

The design of a software system which can implement the multilevel branching of the cross-coupled processing-generating goal-seeking hierarchy shown in Figure 16 is conceptually straight forward. At each level of the generating hierarchy, the H function represents a table, or list, of states and $\underline{S} \rightarrow \underline{P}$ state mappings, one of which is selected by every possible combination of input vectors $\underline{C} + \underline{F}$. At each tick k the selected state mapping H computes an $\underline{P}^k = H(\underline{S}^{k-1})$ *which provides the input to other* modules in the hierarchy at the next time tick. In other words, the input $\underline{C} + \underline{F}$ constitutes an address, or pointer, to a node which contains either the output $\underline{P}$ itself or a procedure for computing $\underline{P}$. This is illustrated in Figure 18.

One method of implementing H modules is to define each state mapping in an H module as a production rule. For example the simple task FETCH (X) is defined by the state table in Figure 19. Note that the lefthand side of the state table contains a command type defined by the input command vector $\underline{C}$, a state defined by self-feedback of the k-1 output U, and flags (or recognized conditions) from the external feedback vector $\underline{F}$.

The right hand side of the state table defines (or points to procedures which define) the output vector $\underline{P}$ which becomes input for other H modules at the next clock tick. The g parameter contained on the right hand side of the state table is sent to the G module (or the M module if one exists) at the corresponding level of the sensory-processing module. The g parameter output at time k selects the G function for the feedback computed at time k+1. For example, in Figure 19, when the g output is g_2, the sensory-processing module is instructed to evoke the G function which computes the orientation of X. The flag bits then indicate whether ORIENTATION (X) > 0 or ORIENTATION (X) $\leq$ 0. When the g output is g_1, the G function which computes the distance to X is evoked.

The entire library of procedures in an H module, and the state table for accessing them is illustrated in Figure 20.

We can now describe a procedure which is executed by the H module at each clock tick. At each time tick k the left hand side of the state table is searched for an entry corresponding to the input $\underline{C}^{k-1}$ $\underline{F}^{k-1}$. If an entry is found, the i pointer is set to that location and $\underline{P}^k$ is used as a pointer to a procedure H which computes an output $\underline{P}^k = H(S^{k-1})$ If no entry can be found the pointer is set to an error condition and a procedure is evoked to output the appropriate failure activities. In most cases a failure condition will output a STOP command to the H module below and a failure flag to the sensory processing G module.

Each entry in the tables of Figures 19 and 20 represent an IF/THEN rule, or production. With this construction it becomes possible to define behavior of arbitrary complexity. An ideal task performance can be defined in terms of the list of events which must take place during the ideal performance. Deviations from the ideal can then be incorporated by simply adding the deviant conditions to the left hand side of the tables and the appropriate action to be taken to the right hand side. Any condition not explicitly covered by the table results in a general failure routine being executed.

Each IF/THEN rule is thus a modular chunk of knowledge which defines a particular condition of the world and what the appropriate response should be. The stringing together of such chunks into H functions for task decomposition or G functions for pattern recognition can then readily be accomplished. New conditions can easily be added and existing rules easily modified. We feel that this has many benefits for teaching industrial robots to perform new tasks and respond to new sensory data.

MICROCOMPUTER NETWORK IMPLEMENTATION

In our laboratory at NBS, we are in the process of implementing the type of cross-coupled measurement/control hierarchy described above. We have chosen to implement the hierarchy in a network of microcomputers because we believe this to be the best way to achieve low cost and upward compatibility.

However, such a logical architecture can be implemented on a minicomputer. In fact, the first version of the NBS hierarchical control system was implemented on a PDP 11/45 [7].

The present system is a network of microcomputers with the architecture shown in Figure 21. Time is sliced into 20 millisecond increments. At the beginning of each increment each logical module reads its set of input values from the appropriate locations in common memory. It then computes its set of output values which it writes back into the common memory before the 20 millisecond interval ends. If a logical module takes longer than the 20 milliseconds to compute an output, an on-board protocol procedure causes the processor to get back in synchronization with the reset pulse before writing out the results to common memory. The process then repeats.

Each logical module is thus a state machine whose output depends only on its present inputs and its present internal state. None of the logical modules admit any interrupts. There is only the reset-sync pulse that signals the beginning and end of the 20 millisecond computation intervals. This simple modular structure enormously simplifies the writing and debugging of software.

THE NBS VISION SYSTEM

The sensory side of the NBS hierarchical control system contains a vision system which uses active illumination to obtain depth information (8). A plane of light is generated by a photoflash tube and a cylindrical lens. This plane is projected into the field of view of a solid state 128x128 automation camera such that the distance to an illuminated surface can be directly computed by simple trigonometry. This camera and flash unit are fixed to the wrist of the robot manipulator.

The control hierarchy activates the vision system at specific points in a particular task execution. The control hierarchy also tells the vision software what type of object to expect and approximately how far away the object is expected to be. The vision software uses this information to select appropriate values for flash intensity and threshold and appropriate software algorithms for processing the visual data.

The vision processing modules either confirm the existence of the expected object and tell the control system where to move to approach it, or report that the expectation was incorrect.

At present, the NBS vision system interfaces with the control system primarily at the primitive action level for computing range and position of grip points and at the elemental move level for computing part orientation and approach paths. However, we are now in the process of adding new capabilities for part recognition at the simple task level.

COMMON MEMORY DATA TRANSFER

All communications of data from one module to another in the NBS hierarchical control system take place via a common memory "mail drop" system as shown in Figure 21. This system has a disadvantage in that it requires two data transfers to get information from one module to another. However, we believe this disadvantage is far overshadowed by the following advantages:

1. There are no communication protocols between computing modules, because modules do not talk directly to each other. Only one processor is allowed to write into any single location in common memory. In each 20 millisecond time slice, all modules read from common memory before any are allowed to write their outputs back in.

2. The addition of each new state variable requires only a definition of where it is to be located in common memory so that the module which generates it knows where to write it, and the modules which read it know where to look. Thus, new microcomputers can easily be added, logical modules can be shifted from one microcomputer to another, new functions such as safety watchdogs, and even new sensors can be included with limited effect on the rest of the system. As long as the system bus has surplus capacity, the physical structure of the system can be reconfigured with few changes required in the software resident in the logical modules.

3. The common memory always contains a readily accessible map of the current state of the system. This makes it easy for a system monitor to trace the history of any or all of the state variables, to set break points, and to reason backwards to the source of program errors or faulty logic. This is extremely important in a sophisticated, real-time, sensory-interactive system in which many processes are going on in parallel at many different hierarchical levels.

FACTORY CONTROL HIERARCHY

NBS has developed an integrated factory concept utilizing the theory of hierarchical control discussed above as well as the experience derived from the microcomputer network implementation. Figure 22 shows the organizational layout. The flow of control during execution is down the center. Orders are entered at the top. Those orders call up process plans which have been entered in the data base on the right. That process plan data base is hierarchically structured so that at the top there is only the name of the process plan. This name is sent to the cell control. The cell control computer accesses the data base which calls in the sequence of steps (ie. the program) that is the process plan at the cell level. Each command in this program is passed in sequence to the next level down which is a work station. As each cell-level command enters the work station, it is the name of a process plan for the work-station. The work station then goes to its data base and calls up the sequence of instructions required to decompose that process plan for the robot or for the machine tool.

The data base on the right also contains part description data such as materials, dimensions, and tolerances. A third section of the right hand data base is the dynamic data related to feeds and speeds which may be changed as a result of sensed conditions in the factory environment.

On the left is a second data base which is also divided into three parts. On the far left is a management information and control data base. Entries or queries to and from this data base enable management to monitor and manage the whole factory by setting priorities, or optimizing for various parameters. The center section of the left hand data base contains the status of each machine tool and robot in the plant: what program is it running, what step in the program, how long in that step, what part is it operating on, etc. The right hand section of this data base contains the status of each part in progress: where is it, what is its position and orientation, what operations have been performed on it, quality control information, etc.

There is a set of computers called feedback processors that operate on each level of this data base and extract out the information needed at the next higher level. As in the microcomputer robot control network, information is passed from one level to another, and from one computing module to another through the data base which serves as a common memory.

Thus, the system is completely modular. A new robot or machine tool or even a work-station can be added or deleted with a minimum of impact on the rest of the system. These data bases

contain a complete state description of the entire factory at each instant of time. Activities of various modules and of the variables themselves can be traced and recorded for debugging, analysis, or optimization. The software is also modular. Programs are written in a language specific to that level. Each computing module becomes a state machine which samples its input for command and feedback variables, performs its computations, and writes its output into the data base, and waits for the next computation cycle.

There are, of course, many unresolved issues. A great deal of additional work needs to be done to specify the data bases, the control software, and the computing architecture. However, it is felt that this approach to a hierarchical control structure for an automatic factory is a concept that will not become obsolete in the near future. It is a real-time sensory-interactive hierarchical control system *with sufficient modularity* that complexity of any module can be kept within tolerable limits regardless of the complexity of the overall system.

It should be noted that hierarchical structure outlined in this paper is at present mainly a theoretical construct which NBS developed and is now in the process of implementing and evaluating. In particular, the software implementation described in Figures 18-20 has not yet been reduced to code. The control structure in Figure 22 is still only in the planning phase. There are many details not covered here regarding timing and syncronization between the behavior-generating and sensory-processing hierarchies. Most of these issues remain to be resolved. There are also many questions regarding how such a system can best be programmed, and how programs can be edited, compiled, and debugged. Nevertheless, we are confident that the cross-coupled processing-generating hierarchy is a fundamentally correct approach to the problem of making industrial robots increasingly responsive to sensory input. In particular, we feel that the modularity of the hierarchical decomposition, and especially the splitting apart of the sensory-processing function from the behavior-generating function, is an important first step in applying the principles of structured programmming to the control of robots. Whether the final result is a hierarchy implemented on a network of small computers, or one implemeted in appropriately structured software on a single large computer, is not important. What is important is that the control problem be decomposed into subproblems which can be solved in a network of computing modules wherein each module has a clearly defined interface of input and output variables and a clearly defined functional relationship between the input and output. This is the first step in good programming practice. We believe it to be essential to the development of sophisticated sensory-interactive control systems for robots and automated factories.

REFERENCES

1. Nilsson, N. J.: Problem-Solving Methods in Artificial Intel ligence, McGraw-Hill, New York, 1871.

2. Albus, J.: Mechanisms of Planning and Problem Solving in the Brain, Mathematical Biosciences, (in press). 3. Albus, J., A. Barbera, J. Evans, G. VanderBrug: Control Concepts for Industrial Robots in an Automatic Factory, Society of Manufacturing Engineers Technical Paper MS77-745.

4. Tinbergen, N.: The Study of Instinct, Clarendon Press, Oxford, 1951.

5. Manning, A.: An Introduction to Animal Behavior, Addison-Wesley, Reading, MA, 1972.

6. von Holst; E. and H. Mittelstaedt: Das Reafferenzprinzip, Die Naturwissenschaften 20:464-476, 1950.

7. Barbera, A.: An Architecture for a Robot Hierarchical Control System, National Bureau of Standards Publication 500-23, 1977.

8. VanderBrug, G., J. Albus, and E. Barkmeyer: A Vision System for Real-Time Control of Robots, Proceedings 9th International Symposium on Industrial Robots, Washington, DC, March, 1979.

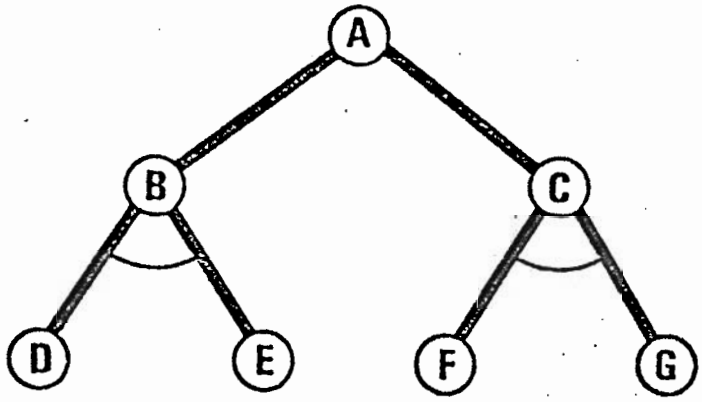

FIGURE 1. An AND/OR graph.

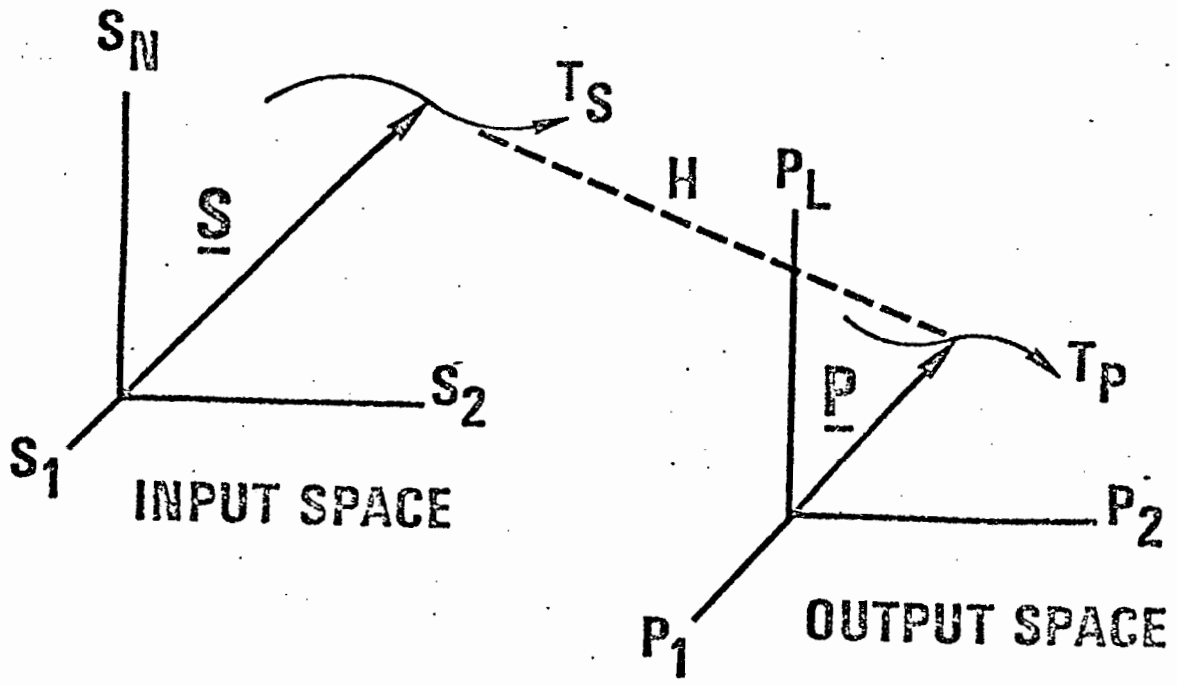

FIGURE 2. The function H maps the trajectory T_s into the trajectory T_p.

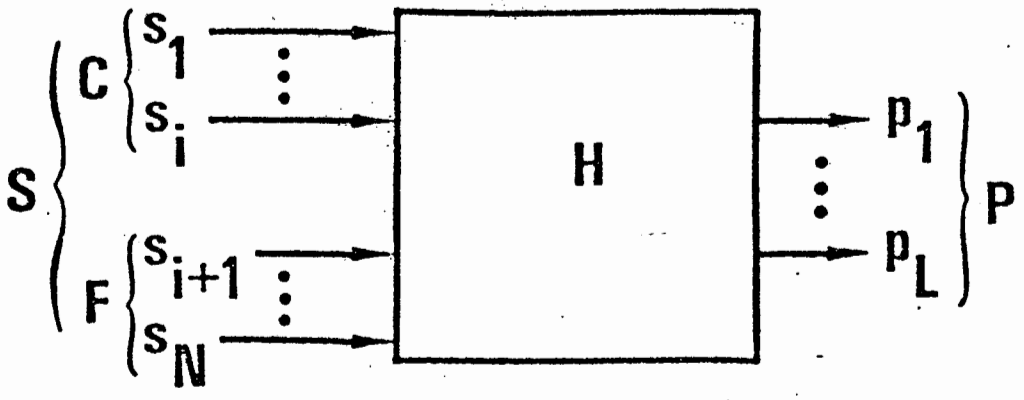

FIGURE 3. The input vector S is divided into two parts C and F such that S = C+F.

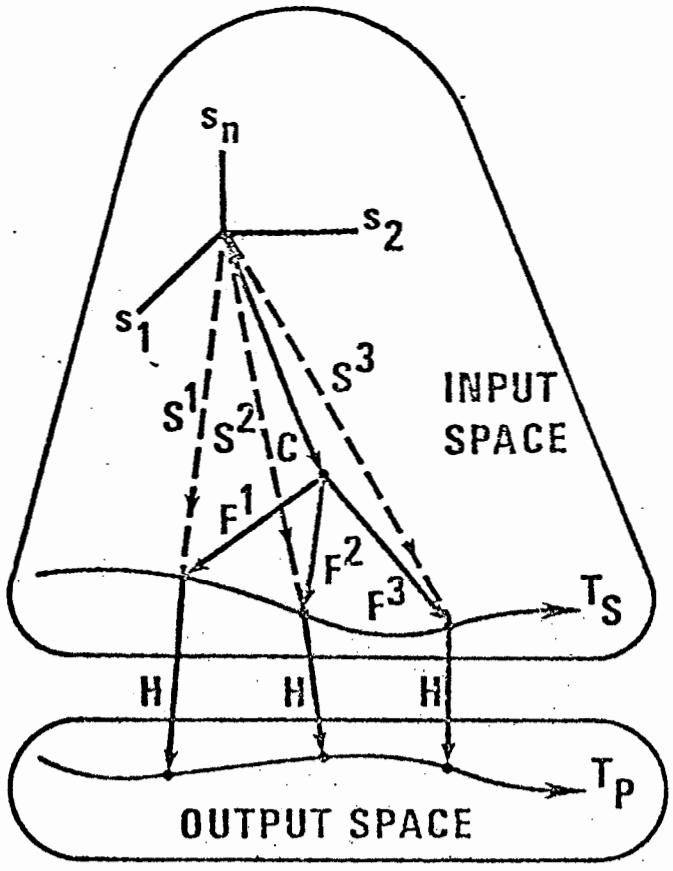

FIGURE 4. A stationary C vector establishes a setpoint, and as time progresses the feedback vector varies from $\underline{F}^1$ to $\underline{F}^2$ to $\underline{F}^3$. The S vector thus traces out a trajectory T_s. The H operator computes an output P for each input S and so produces an output trajectory T_p. The result is that the input command C is decomposed into a sequence of output subcommands $\underline{P}^1$, $\underline{P}^2$, $\underline{P}^3$.

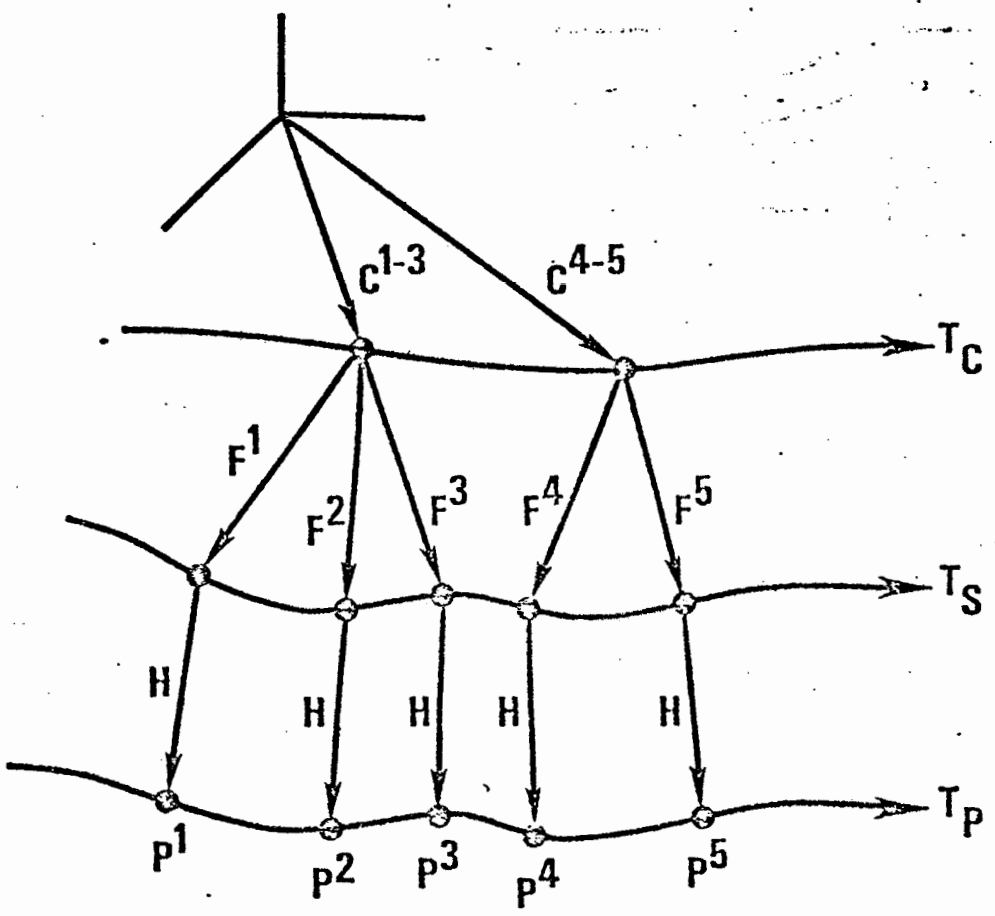

FIGURE 5. If the command vector $\underline{C}$ also changes from time to time, it will trace out a trajectory T_c.

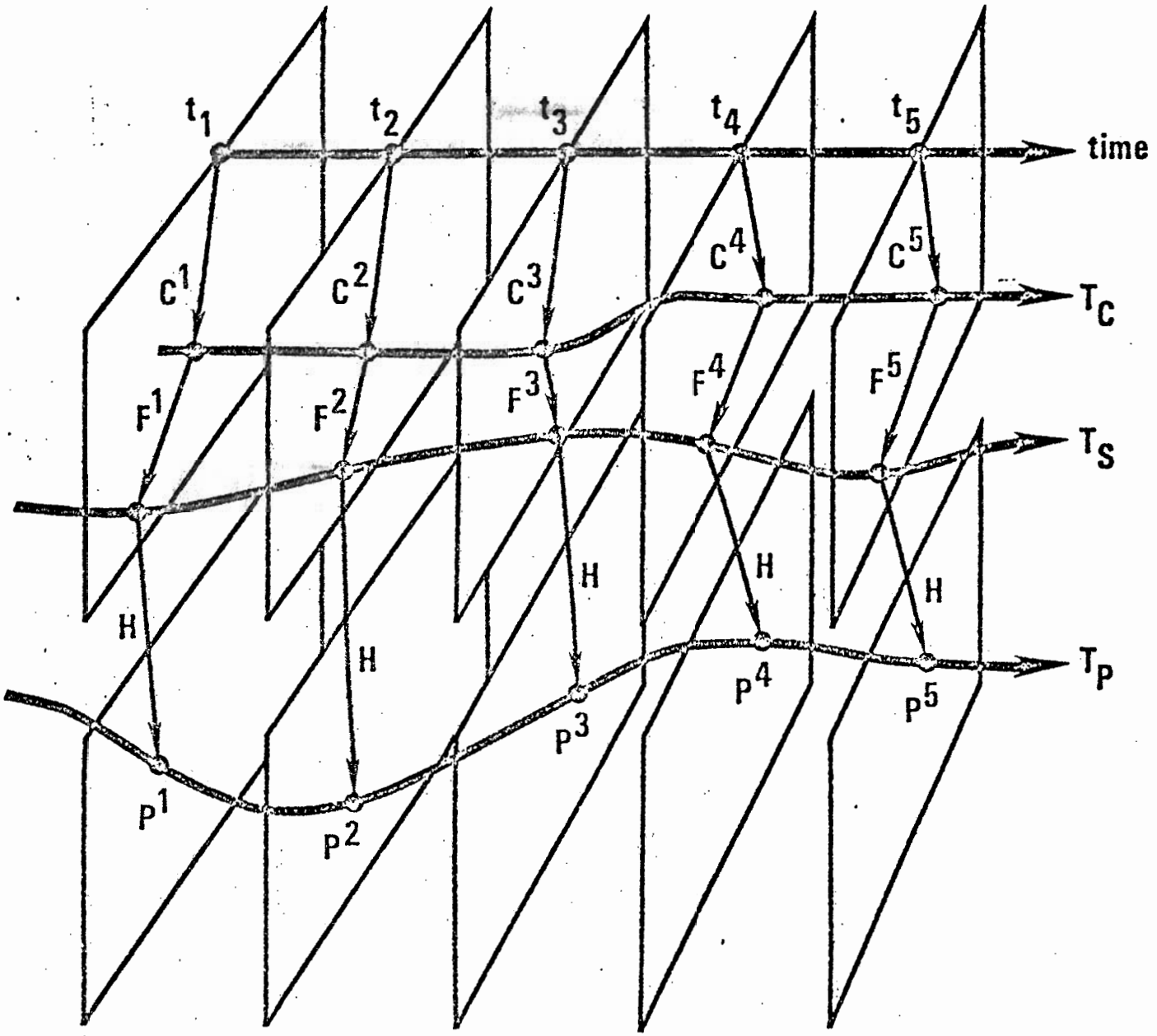

FIGURE 6. When time is represented explicitly, the vectors and trajectories of Figure 5, become as shown here. In this example, the C vector remains constant from time t = 1 to t = 3 and then jumps to a new value for t = 4 and t = 5.

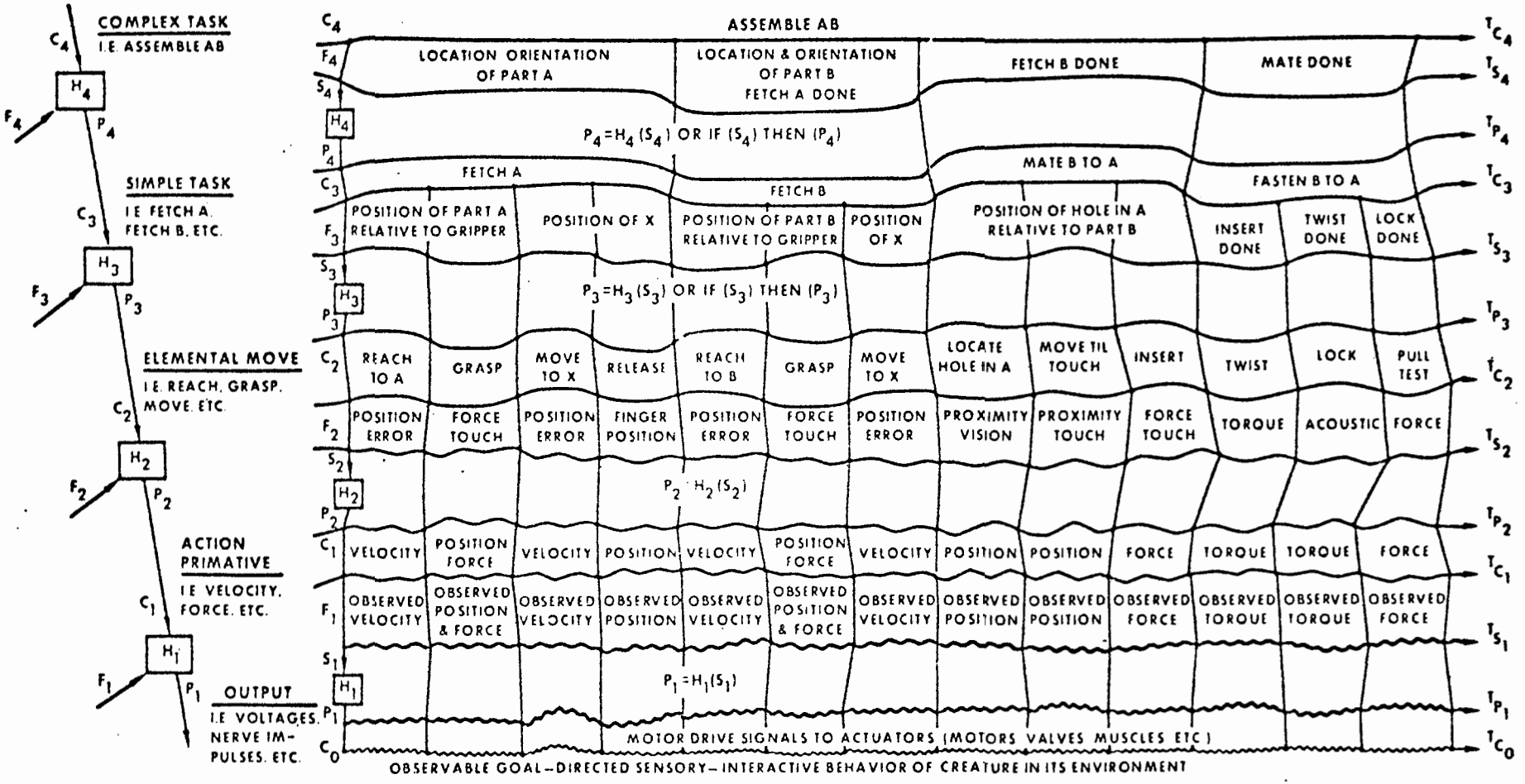

FIGURE 7. A hierarchy of H operators produces sensory-interactive goal-directed behavior.

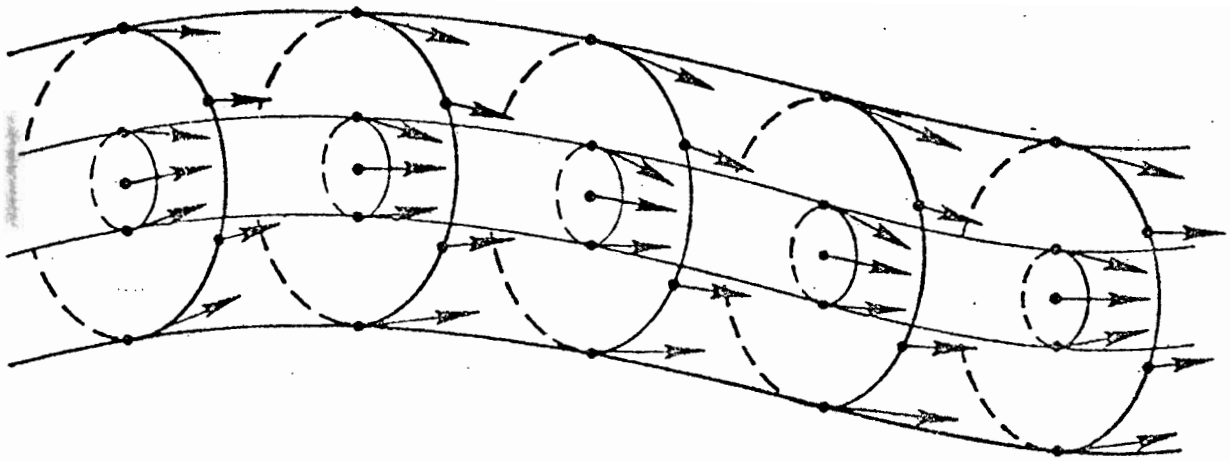

FIGURE 8. Around each trajectory representing an ideal task performance there exists an envelop of close-to-ideal trajectories which correspond to successful, but not perfect, task performance. If the H functions are defined throughout these envelops so as to drive the system back toward the ideal whenever it deviates, then the trajectory will be stable and task performance can be successful despite perturbations and unexpected events.

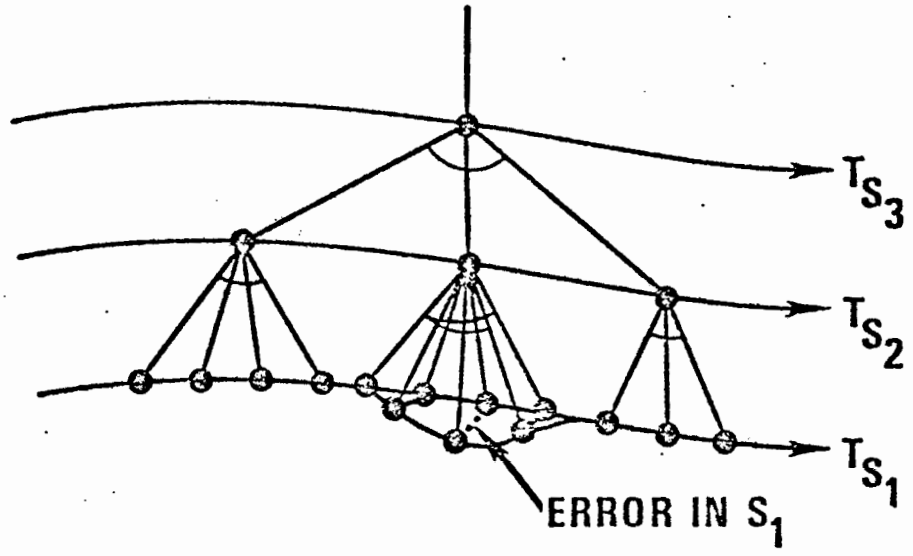

FIGURE 9. If the H function at the lower levels are sufficiently well defined, small perturbations from the ideal performance can be corrected by low level feedback without requiring any change in the command from higher levels.

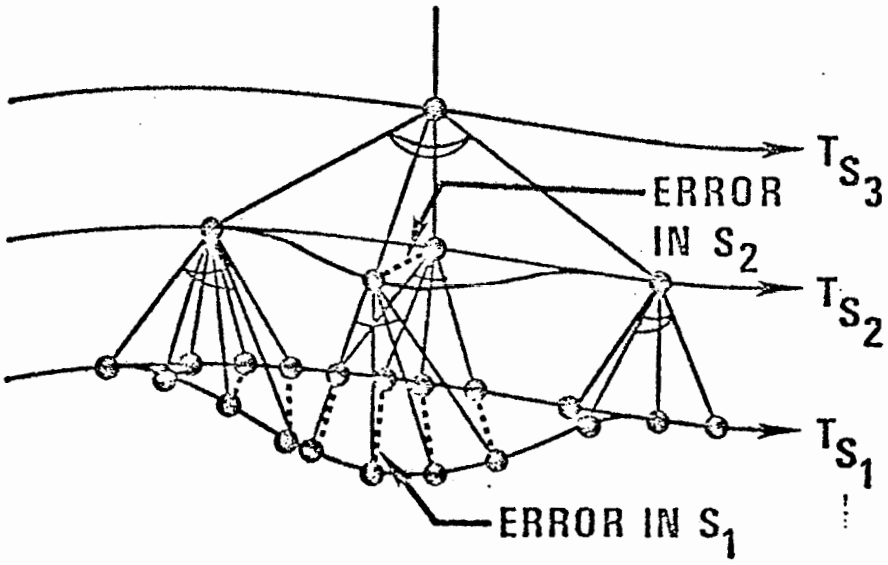

FIGURE 10. If the lower level H functions are not adequately defined, or if the perturbations are too large for the lower level to cope, then feedback to the higher levels produces changes in the task decomposition at a higher level. The result is an alternative strategy.

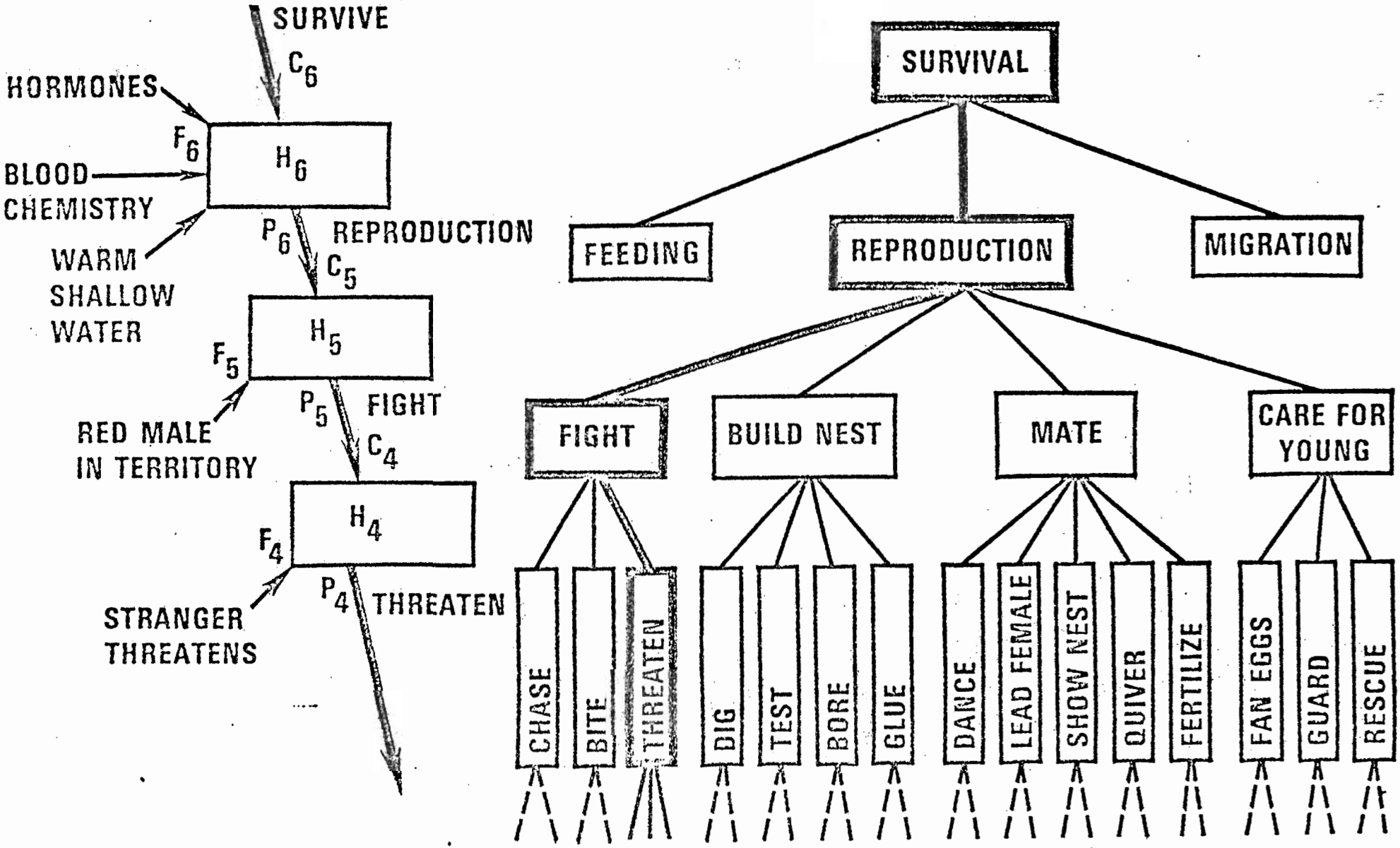

FIGURE 11. The command and control hierarchy proposed by Tinbergen to account for the behavior of the male three-spined stickleback fish. The heavy line indicates the particular type of behavior vector actually selected by the feedback shown at the various levels of the hierarchy on the left. This figure represents a snap-shot in time corresponding to one of the 2-dimensional surfaces shown in Figure 6

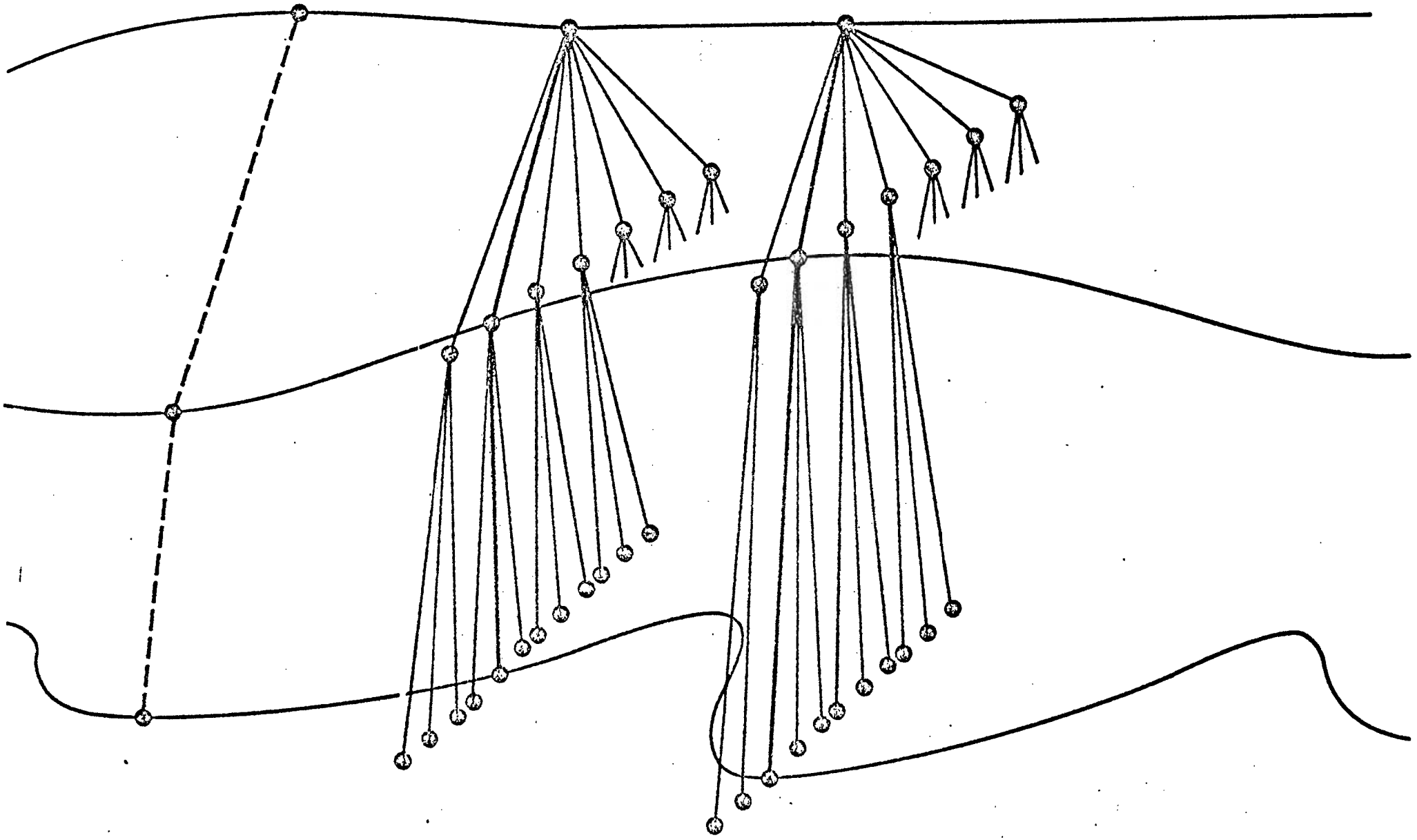

FIGURE 12. A set ot T_p trajectories in which there is opportunity for branching at many points in time. If behavior can be modified by feedback at many different levels and in many different ways, it appears to be adaptive and flexible. If there are only a few branch points, with only a few alternative actions available at each branch, behavior will appear stereotyped.

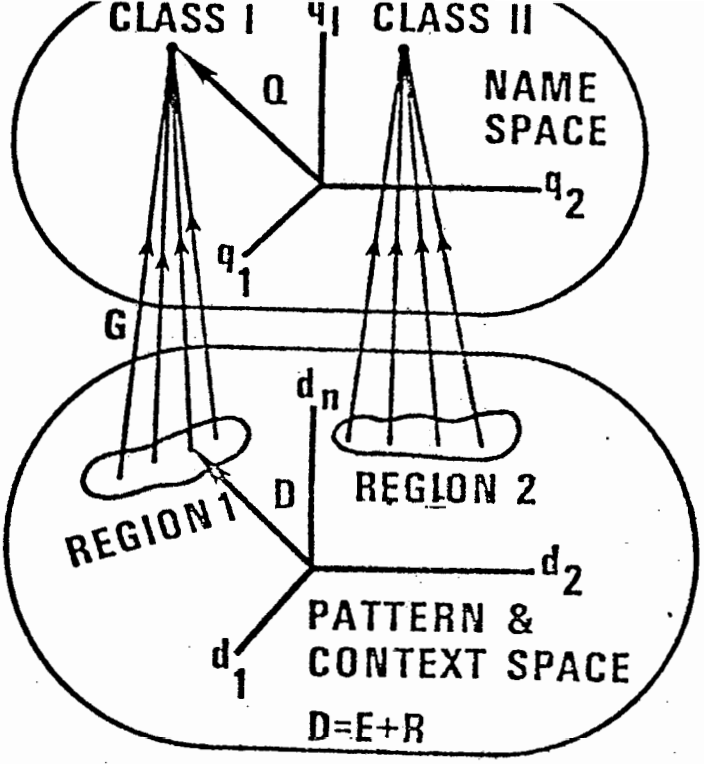

FIGURE 13. The D vector is composed of sensory variables E and context variables R. The function G recognizes the existence of a D vector in a particular region of pattern+context space by outputting a Q vector which is the name of that region.

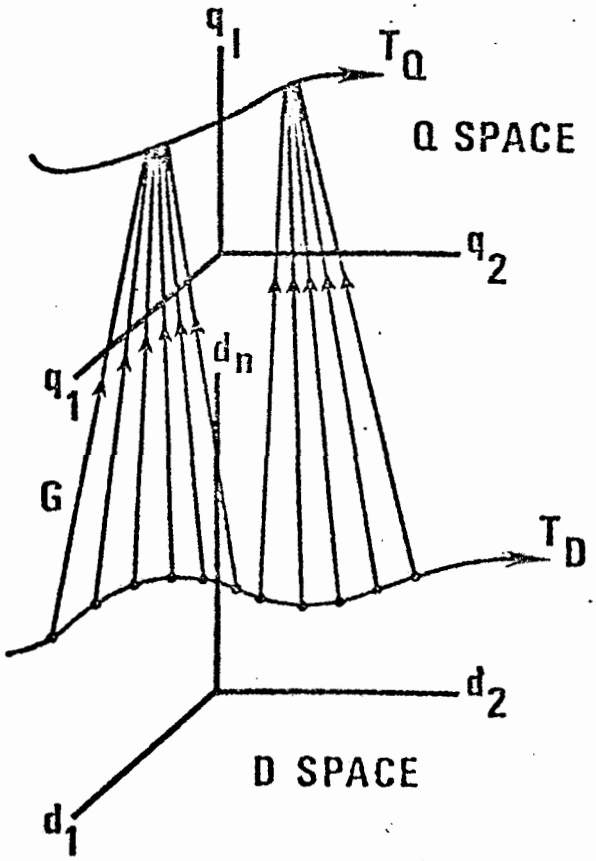

FIGURE 14. A time varying D vector traces out a trajectory T_D which represents a sensory experience T_E taking place in the context T_R. A section of a T_D trajectory which maps into a small region of Q space corresponds to the recognition of an extended temporal pattern as a single event.

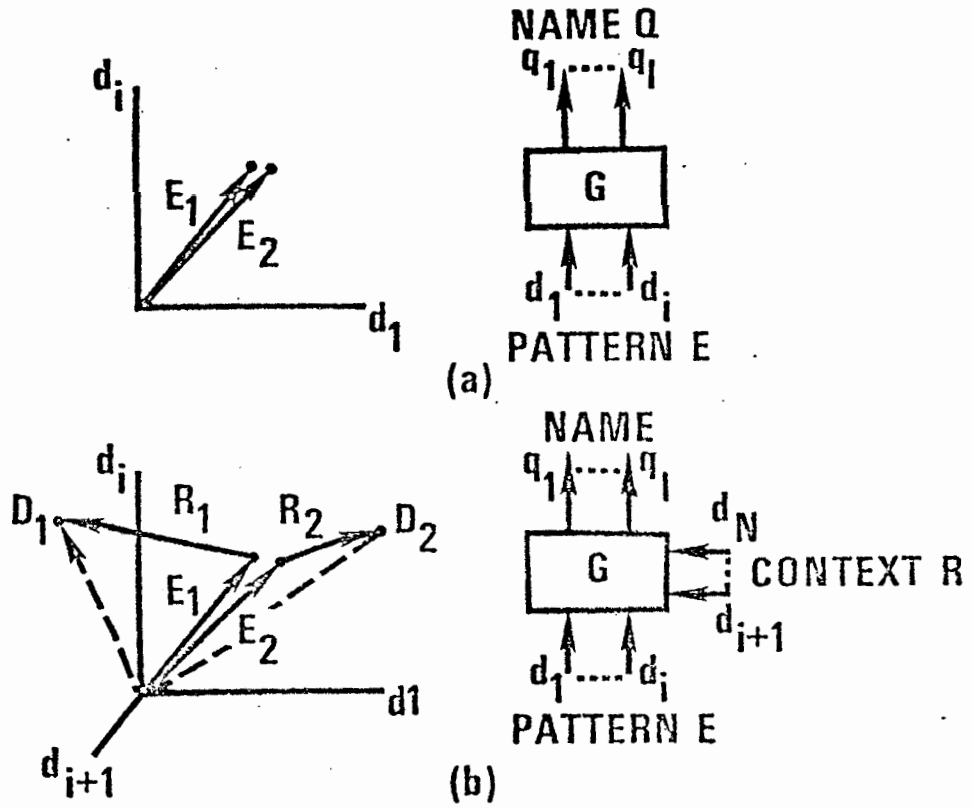

FIGURE 15. In (a) the two pattern vectors $\underline{E}_1$ and $\underline{E}_2$ are too close together in pattern space to be reliably recognized (i.e. named) as in different classes. In (b) the addition of context $\underline{R}_1$ to $\underline{E}_1$ and $\underline{R}_2$ to $\underline{E}_2$ makes the vectors $\underline{D}_1$ and $\underline{D}_2$ far enough apart in pattern+context space to be easily recognized as in separate classes.

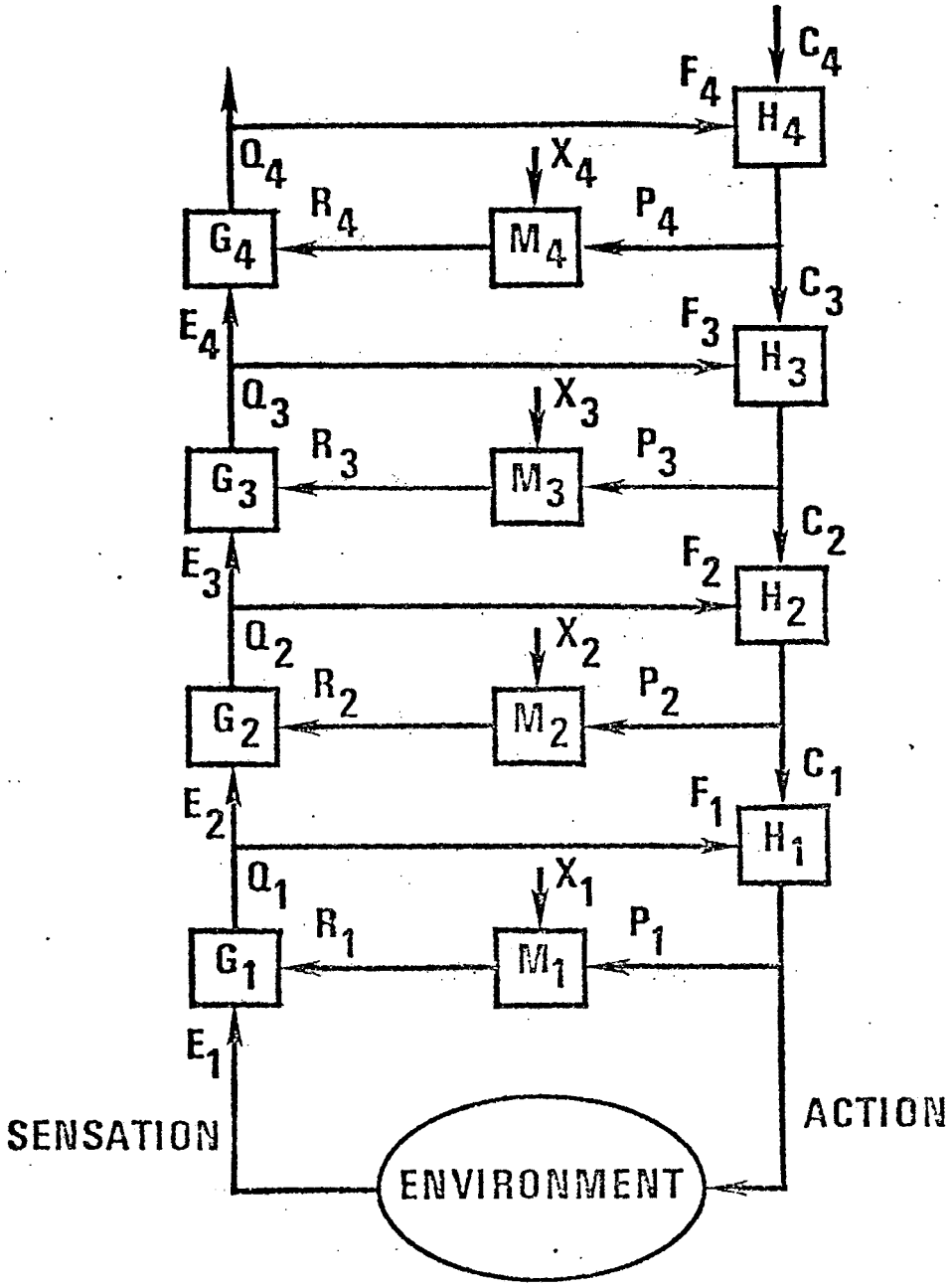

FIGURE 16. A cross-coupled processing-generating hierarchy. The M_i modules remember sensory experiences which occur in association with specific activity in the generating hierarchy ($\underline{P}_i$) and other sensory modalities ($\underline{X}_i$). The M_i modules thus learn a set of internal expectations (i.e. a predictive model) of the external world as seen through the sensory input channels.

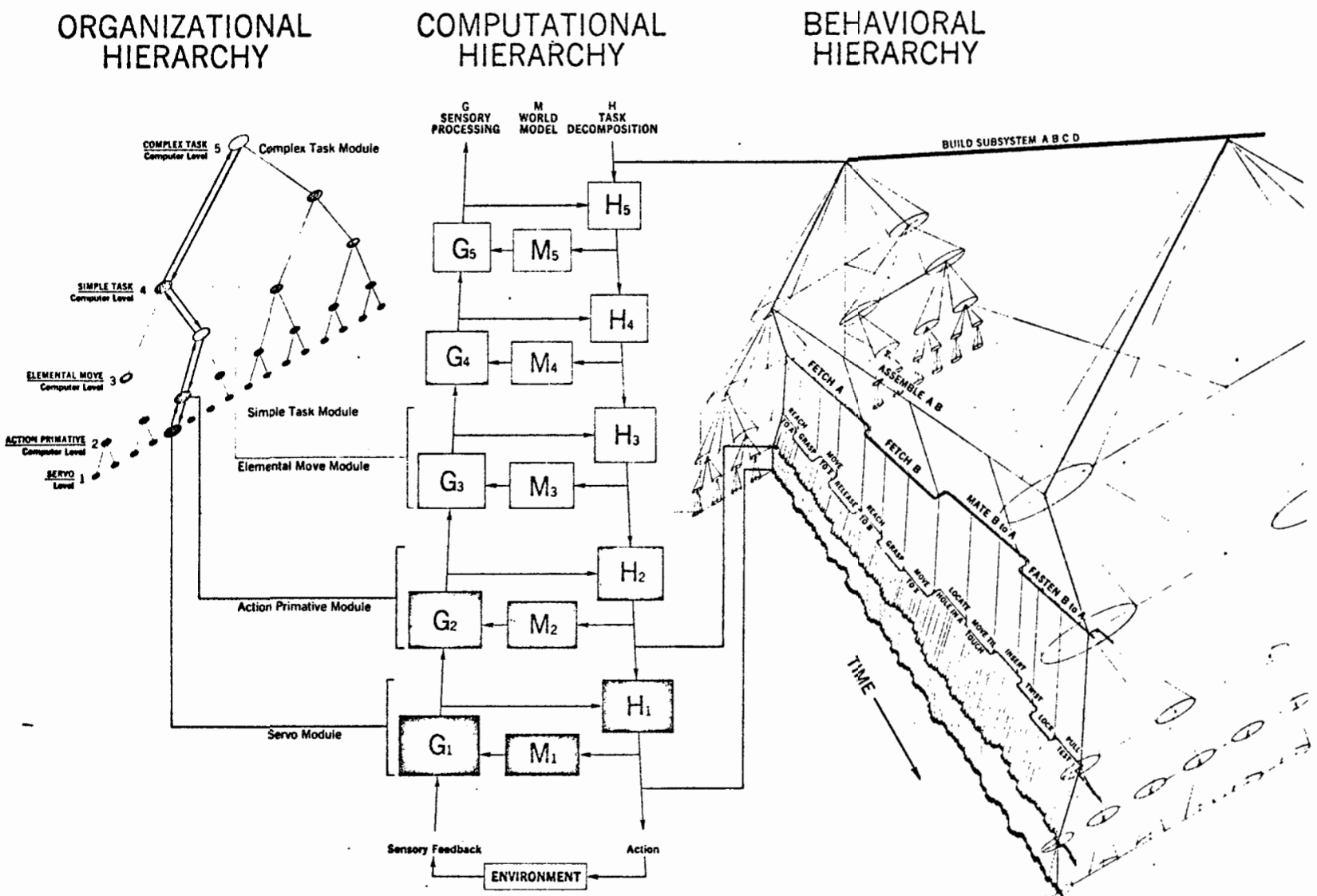

FIGURE 17. Three aspects of a sensory-control system.

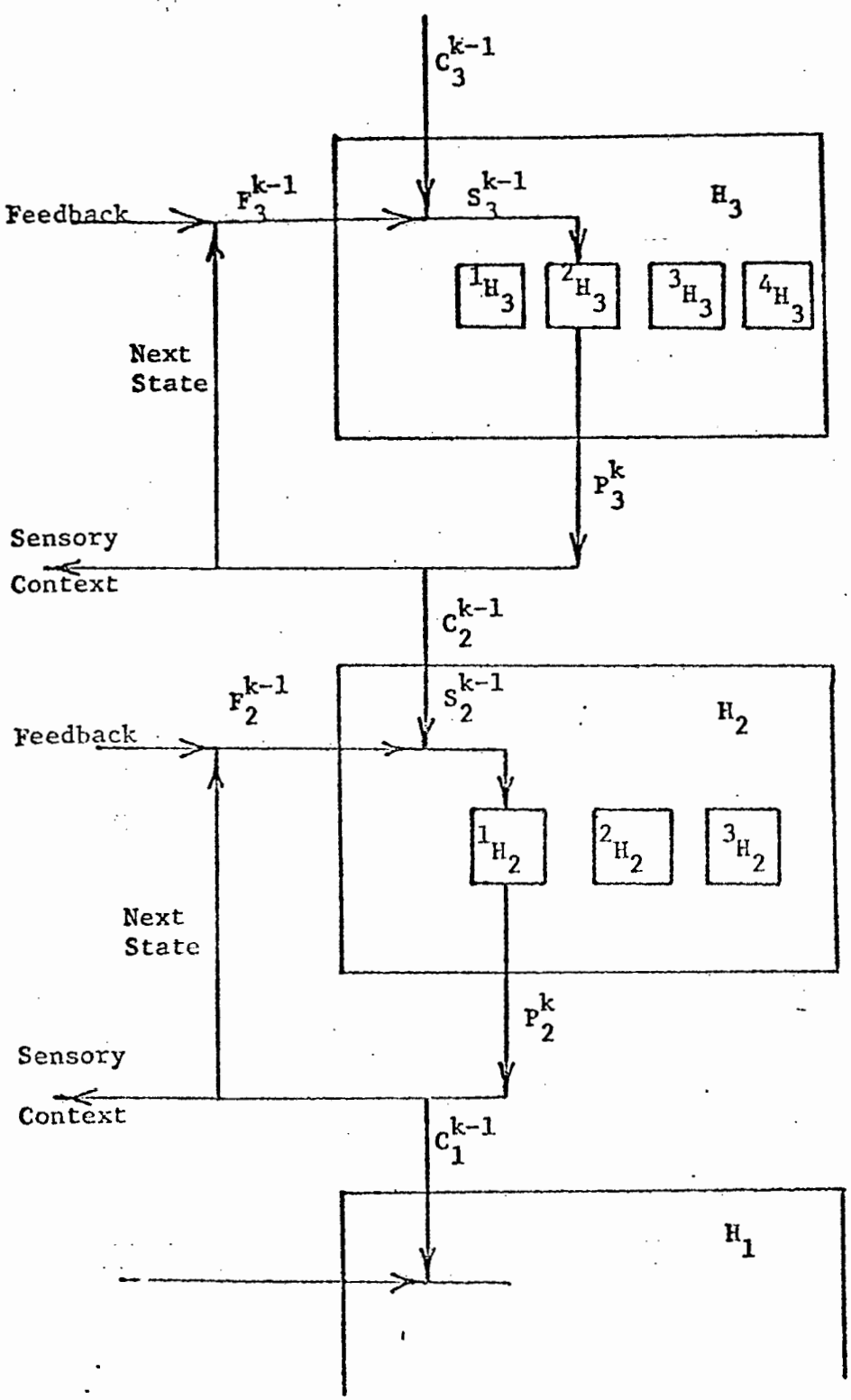

FIGURE 18. The input $\underline{S}_i^{k-1} = \underline{C}_i^{k-1} + \underline{F}_i^{k-1}$ constitutes an address, or pointer, to a node $S_{i_{H_i}}^{k-1}$ which contains either the output $\underline{P}_i^k$ itself or a procedure for computing $\underline{P}_i^k$.

C^{k-1}	F^{k-1}		P^k		
Command (Argument)	State U^{k-1}	Feedback Flags	Output	Next State U^k	Sensory Context
FETCH(X)	0	-	SEARCH FOR(X)	1	g_1
FETCH(X)	1	Dist(X) = ?	SEARCH FOR(X)	1	g_1
FETCH(X)	1	Search fail	STOP REPORT FETCH-FAIL	0	g_o
FETCH(X)	1	Dist(X)> a	GO TO (Pos(X))	1	g_1
FETCH(X)	1	Dist(X) $\leq$ 0	ORIENT ON(X)	2	g_2
FETCH(X)	2	Orient(X)> 0	ORIENT ON(X)	2	g_2
FETCH(X)	2	Orient(X) $\leqq$ 0	GRASP(X)	3	g_3
FETCH(X)	3	Touch < T	GRASP(X)	3	g_3
FETCH(X)	3	Touch $\geqslant$ T	GO TO JIG(X)	4	g_4
FETCH(X)	4	Dist(JIG)> 0	GO TO JIG(X)	4	g_4
FETCH(X)	4	Dist(JIG) = 0	INSERT(X)	5	g_5
FETCH(X)	5	Force < 0	INSERT(X)	5	g_5
FETCH(X)	5	Force > 0, Insert done	RELEASE	6	g_6
FETCH(X)	6	Releasing	RELEASE	6	g_6
FETCH(X)	6	Release done	STOP Report FETCH-DONE	0	g_6
FETCH(X)	-	Subtask fail	STOP Report FETCH-Fail	0	g_o

FIGURE 19. One method for implementing each of the S → P mappings of Figure 18 is through a state table such as shown here.

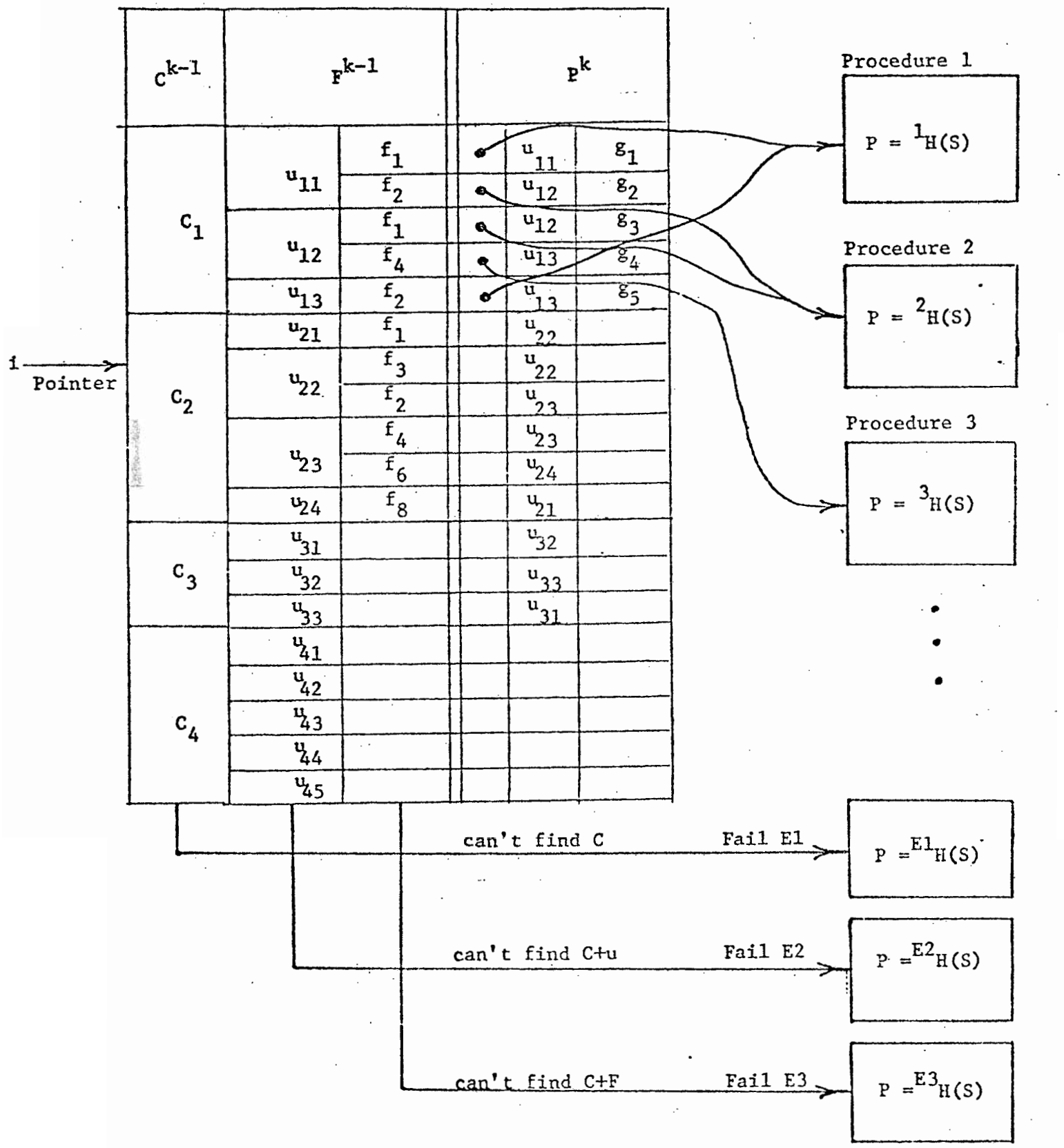

FIGURE 20. The entire library of procedures in an H module can be represented as an extended state table.

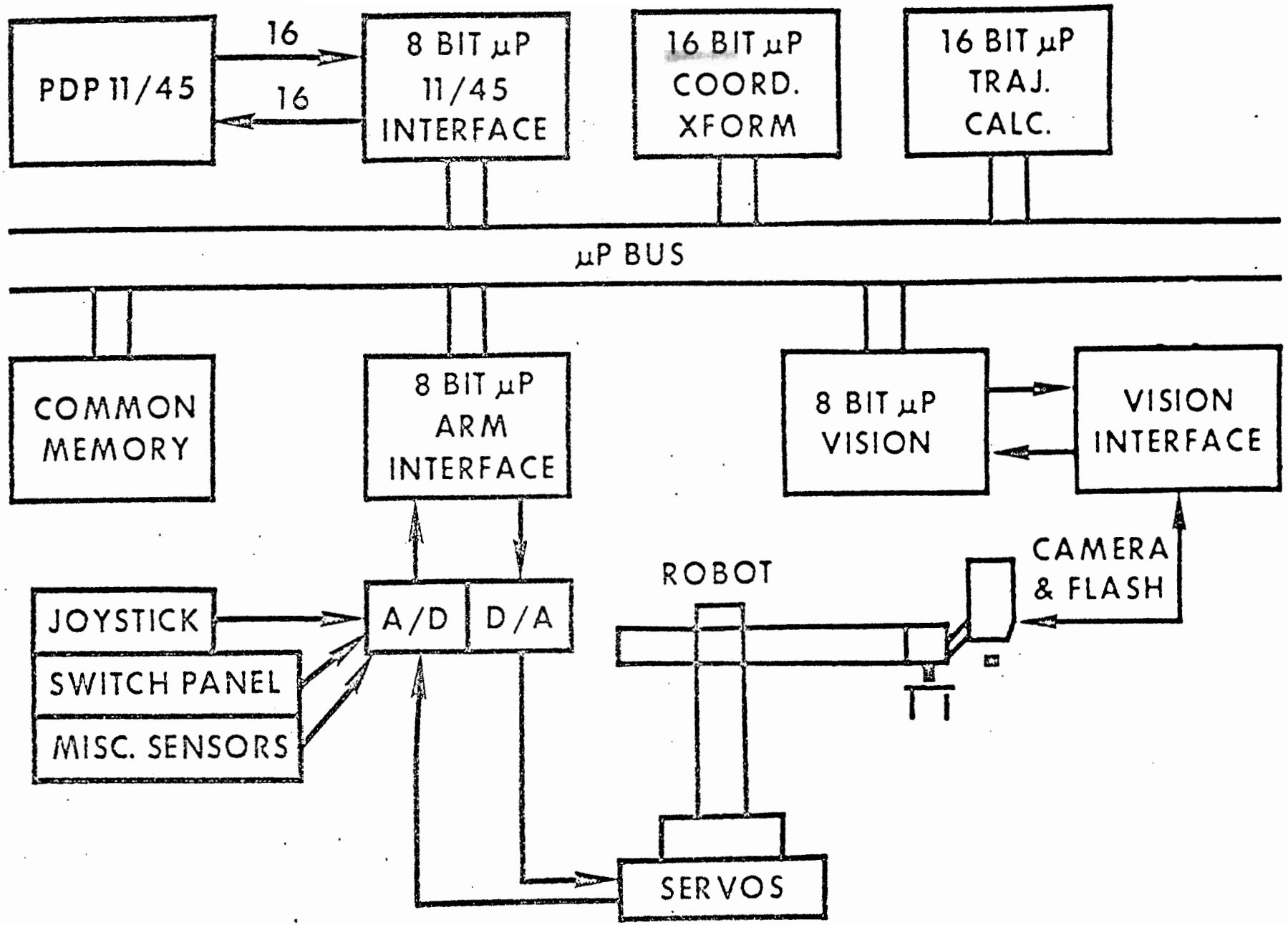

FIGURE 21. The NBS microcomputer network architecture for implementing a hierarchical robot control system.

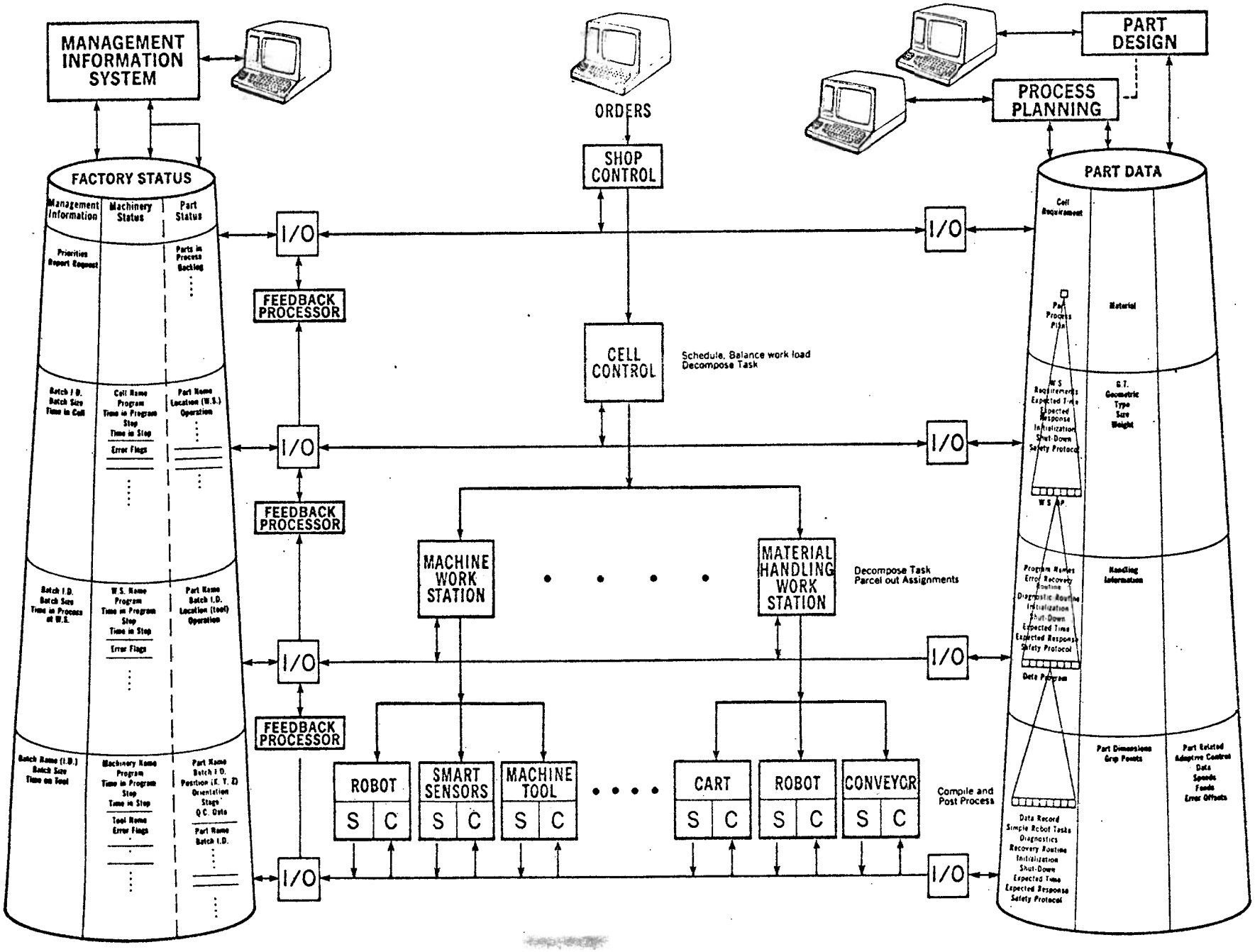

FIGURE 22. A hierarchical architecture for a factory control system.